BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem informasi merupakan sekumpulan elemen yang saling terhubung atau berkaitan untuk tujuan tertentu, proses pengolahan data mentah menjadi informasi secara tersistem dapat menghasilkan informasi yang lebih sederhana dan mudah digunakan oleh pengguna (Saputra, *et al.,* 2021).

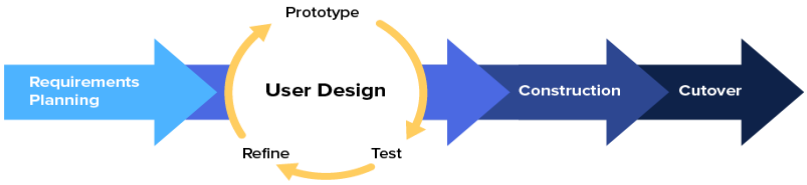
2.2 Metode Pengembang Sistem

Metode pengembang sistem merupakan metode yang digunakan sebagai alur proses dalam pengembangan, sehingga penelitian dapat di kembangkan sesuai tahapan dari metode pengembang sistem.

2.2.1 *Rapid Application Development* (RAD)

Metode yang berfokus pada pengembangan aplikasi secara cepat, melalui pengulangan dan *feedback* berulang-ulang. RAD diajukan oleh IBM pada tahun 1980 sampai 1990-an, ketika permintaan terhadap aplikasi semakin meningkat. Dengan banyaknya *demand*, orang-orang di dunia IT harus mencari solusi untuk memenuhi permintaan tersebut. Metode ini merupakan semacam cikal bakal *agile project management*, karena bisa mengikuti *pace* bisnis yang terus berkembang dan juga kebutuhan pasar yang terus meningkat. Pengembangan software pada umumnya seperti waterfall model membutuhkan perencanaan yang terbilang cukup kaku. Klien atau pelanggan seakan ‘dipaksa’ untuk menyetujui banyak hal di awal, tetapi mereka tidak bisa melihat proses pembuatannya (Rosa and Shalahuddin, 2019).

Keuntungan utama menjalankan *rapid application development* adalah jangka waktu pengembangan lebih cepat. Hal ini dikarenakan *feedback* dari pelanggan cepat didapatkan dan semua perubahan yang dilakukan akan sesuai hasil tersebut. Akan tetapi, salah satu kekurangan RAD adalah kamu membutuhkan tim berisikan *developer* yang benar-benar memiliki *skill* tinggi dan juga metode ini hanya bisa digunakan untuk proyek yang bisa termodulasi.



Gambar 1.1 *Rapid Application Development* (RAD)
Sumber: (Rosa and Shalahuddin, 2019)

1. Kelebihan Model RAD

   Kelebihan metodologi RAD menurut Marakas (2006):

   a. Penghematan waktu dalam keseluruhan fase projek dapat dicapai.
   b. RAD mengurangi seluruh kebutuhan yang berkaitan dengan biaya projek dan sumberdaya manusia.
   c. RAD sangat membantu pengembangan aplikasi yang berfokus pada waktu penyelesaian projek.
   d. Perubahan desain sistem dapat lebih berpengaruh dengan cepat dibandingkan dengan pendekatan SDLC tradisional.
   e. Sudut pandang user disajikan dalam sistem akhir baik melalui fungsi-fungsi sistem atau antarmuka pengguna.
   f. RAD menciptakan rasa kepemilikan yang kuat di antara seluruh pemangku kebijakan projek.

2. Kelemahan Model RAD

   Kelemahan pada pengembangan tersebut dapat dilihat berdasarkan kesesuaian pengembangan yang dilakukan, berikut adalah kelemahan metode pengembang sistem RAD :

   a. Dengan metode RAD, penganalisis berusaha mepercepat projek dengan terburu-buru.
   b. Kelemahan yang berkaitan dengan waktu dan perhatian terhadap detail. Aplikasi dapat diselesaikan secara lebih cepat, tetapi tidak mampu mengarahkan penekanan terhadap permasalahan-permasalahan perusahaan yang seharusnya diarahkan.

c. RAD menyulitkan *programmer* yang tidak berpengalaman menggunakan prangkat ini di mana *programmer* dan *analyst* dituntut untuk menguasai kemampuan-kemampuan baru sementara pada saat yang sama mereka harus bekerja mengembangkan sistem

2.2.2 Tahapan Penelitian

Tahapan dalam penelitian sebagai langkah-langkah penelitian yang harus dikerjakan, berikut adalah tahapan penelitian Rapid Application Development.

1. Tahap *Requirements Project*

   RAD dimulai dengan menentukan kebutuhan sebuah proyek (*project requirements*). Pada tahap ini, tim perlu menentukan kebutuhan yang ingin dipenuhi dari sebuah proyek. Kebutuhan ini tidak perlu spesifik. Tapi, sifatnya benar-benar umum dan jumlahnya bisa banyak. Baru dari situ, tim akan menentukan mana kebutuhan yang perlu diprioritaskan. Setelah mendapatkan kebutuhan yang jelas, barulah tim menentukan hal-hal yang lebih detail. Misalkan seperti tujuan, timeline, dan budget yang diperlukan.

2. Tahap Membuat *Prototype*

   Hal yang selanjutnya dilakukan adalah membuat prototype. Developer secepat mungkin akan membuat prototype dari aplikasi yang diinginkan. Lengkap dengan fitur dan fungsi yang berbeda-beda. Tujuannya, sekadar untuk mengecek apakah *prototype* yang dibuat sudah sesuai dengan kebutuhan klien. Meski begitu, tahap ini bisa saja dilakukan berulang-ulang. Kadang juga melibatkan user untuk testing dan memberikan feedback. Proses ini memungkinkan tim mempelajari error yang mungkin muncul ke depannya. Ini berguna untuk mengurangi error dan debugging. Lewat tahapan ini, tim developer memiliki modal untuk membuat aplikasi yang mudah dipakai, stabil, tidak sering error, dan desainnya pun oke.

3. Proses Pengembangan dan Penngumpulan *Feedback*

   Setelah tahu aplikasi seperti apa yang ingin dibuat, developer mengubah *prototype* ke bentuk aplikasi versi beta sampai dengan final. Jadi, bisa dibilang tahap RAD inilah yang cukup intens. Developer terus-menerus melakukan coding aplikasi, melakukan testing sistem, dan integrasi dengan bagian-bagian lainnya. Karena itulah, developer menggunakan tools dan framework yang

mendukung RAD agar cepat. Apalagi proses ini terus diulang sambil terus mempertimbangkan feedback dari klien. Baik itu soal fitur, fungsi, interface, sampai keseluruhan aspek dari produk yang dibuat. Nah, kalau prosesnya berjalan lancar, developer akan melanjutkan ke langkah berikutnya. Yaitu, finalisasi produk atau implementasi. Kalau pun tidak, proses ini kemungkinan akan terus diulang. Pun, kalau apes-apesnya aplikasi tidak tidak menjawab kebutuhan, developer akan kembali ke proses *prototyping*.

4. Tahap *Implementation* (implementasi).
   Merupakan tahap pengujian terhadap aplikasi yang dikembangkan. Tahap ini programmer mengembangkan desain menjadi suatu program kemudian dilakukan proses pengujian untuk memeriksa kesalahan sebelum diaplikasikan.

2.3 *Unified Modelling Language* (UML)

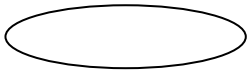
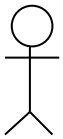
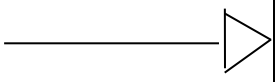
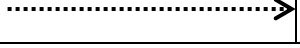
UML *(unified Modelling Language)* adalah bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada UML *(Unified Modelling Language)* (Rosa and Shalahuddin, 2019).

2.3.1 *Use Case Diagram*

Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada Tabel 2.1.

Tabel 1.1 Simbol *Use Case Diagram*

No	Simbol	Deskripsi

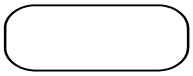
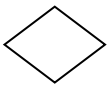
1.		*Usecase* Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case*.
2.		Aktor Aktor seseorang/sesuatu yang berinteraksi dengan yang akan dibuat. diluar sistem informasi. Biasanya dinyatakan menggunakan kata benda
3.		Asosiasi/association merupakan komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
4.		Generalisasi *(generalization)* merupakan hubungan (umum – khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum
5.	<< *Include* >> 	Include berarti use case yang ditambahkan akan dipanggil saat use case tambahan dijalankan.
6.	<<*Extend*>> 	Ekstensi *(extend)* merupakan use case tambahan ke sebuah use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu.

2.3.2 *Activity Diagram*

Activity diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *activity diagram* dapat dilihat pada tabel 2.2 berikut ini :

Tabel 1.2 Simbol *Activity* Diagram

No.	Simbol	Keterangan

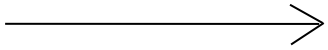
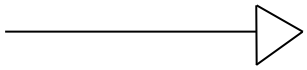
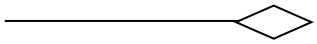
1.		Status awal aktivitas sitem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Percabangan *(Decision)* merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan *(Join)* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas
5.	Nama *swimlane*	Swimlane Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.3.3 *Class Diagram*

Class diagram mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada tabel 2.3.

Tabel 1.3 Simbol *Class* Diagram

No.	Simbol	Deskripsi
1.	Nama_kelas *+Atribute* +Operasi	Kelas pada struktur sistem**.**
2.	Antar Muka/*Interface*  Nama_*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.	Asosiasi / *Asociation* ———————	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan symbol

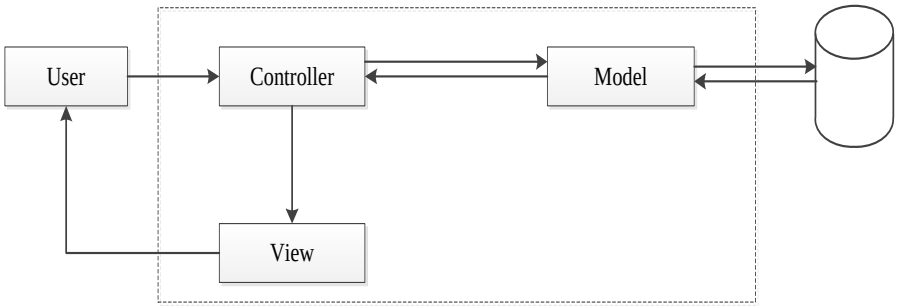
4.	Asosiasi Berarah / *Directed Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan symbol.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Agregasi / *aggregation* 	Relasi antar kelas dengan maksna semua bagian *(whole-part)*

2.4 *CodeIgniter*

CodeIgniter adalah *Framework* untuk bahasa pemprograman PHP, yang dibuat Rick Ellis pada tahun 2006. *CodeIgniter* memiliki banyak fitur yang membantu para pengembang PHP untuk dapat membuat aplikasi secara mudah dan cepat serta memiliki sifat yang fleksibel dapat mengembangkan dalam perangkat *web*, *dekstop* maupun *mobile* (Raharjo, 2018).

Codeigniter adalah sebuah aplikasi gratis yang berupa kerangka kerja untuk membangun website menggunakan bahasa pemprograman PHP (Heru, 2018).

CodeIgniter memilki konsep atau pola *Model-View-Controller* (MVC) sehingga kode-kode dapat di sederhanakan.



Gambar 1.2 Arsitektur MVC
Sumber: (Raharjo, 2018)

Berdasarkan arsitektur tersebut diketahui konsep *Model-View-Controller* yaitu:

1. *Model*

   Model merupakan suatu fungsi yang digunakan mengelola database pada sistem untuk menangani struktur data dari *controller*.

2. *View*

   View merupakan bagian untuk mengelola tampilan dari *website* atau dapat disebut sebagai *user interface* yang diatur bagian *controller.*

3. *Controller*

   Controller merupakan kunci dalam konsep MVC dengan fungsi untuk menghubungkan *model* dengan *view.*

CodeIgniter merupakan sebuah *toolkit* yang ditujukan untuk orang yang ingin membangun aplikasi web dalam bahasa pemrograman PHP. Beberapa keunggulan yang ditawarkan sebagai berikut:

1. CodeIgniter adalah *framework* yang bersifat *free* dan *open source*
2. Memiliki ukuran file yang relatif lebih kecil dibanding *framework* lain
3. Aplikasi yang dihasilkan dapat berjalan cepat.
4. Menggunakan pola desain MVC sehingga satu file tidak banyak berisi kode, hal tersebut menjadikan kode mudah dibaca dan dipahami.
5. Dapat diperluas sesuai kebutuhan

   Dokumentasi penerapan CodeIgniter dapat dipahami dengan mudah.

2.5 Pengujian Sistem *Black Box*

Black Box Testing merupakan pengujian yang dapat dilakukan dengan melakukan pengamatan, pada hasil eksekusi melalui beberapa data uji dan memeriksa fungsional yang terdapat pada perangkat lunak. Jadi dapat kita dianalogikan seperti halnya kita melihat ke dalam kotak hitam, sehingga kita hanya bisa melihat tampilan luarnya saja tanpa kita tau apa yang ada didalam kotak hitam tersebut (Rosa and Shalahuddin, 2019).

Black box testing merupakan pengujian terhadap fungsi sistem yang dilakukan langsung kepada pengguna sistem untuk mengetahui keberhasilan setiap fungsi yang telah diuji (Suryantara, 2017).

Sehingga sama seperti halnya dengan *Black Box Testin*g yang hanya dapat mengevaluasi dari tampilan luarnya dan fungsionalitasnya. Tanpa harus mengetahui apa sesungguhnya yang terjadi dalam proses detilnya. Pada pengetahuan khusus dari struktur kode internal dan pengetahuan pada pemrograman dasar pada umumnya tidak diperlukan untuk *Black Box Testing*. Uji

pada kasus yang dibangun disekitar spesifikasi dan persyaratan, yakni pada aplikasi yang seharusnya dilakukan.

2.6 Penelitian Terdahulu

Penelitian terdahulu bertujuan untuk mendapatkan bahan perbandingan dan acuan. Selain itu, untuk menghindari anggapan kesamaan dengan penelitian ini. Maka dalam kajian pustaka ini peneliti mencantumkan hasil-hasil penelitian terdahulu sebagai berikut:

Oleh Fajar Sarasati, Diah Pradiatiningtyas dan Nani Purwati, (2021). Sistem Informasi, STMIK Nusa Mandiri, Sistem Informasi Akuntasi, Universitas Bina Sarana Informatika. Perancangan E-Bakul Pada Kelompok Wanita Tani Ngudi Rejeki Berbasis Website. Masalah pada penelitian ini yaitu strategi pemasaran yang dilakukan kurang efektif dan efesien, transaksi pemasaran masih melalui via WhatsApp dan datang langsung ke toko dan reseller membutuhkan informasi produk yang lengkap. Metode yang digunakan dalam penelitian ini yaitu metode SDLC (Waterfall) terdiri atas 5 tahapan, tetapi dalam penelitian ini hanya menggunakan 2 tahap yaitu analisis kebutuhan perangkat lunak dan desain. Hasil penelitian yang diperoleh sistem informasi E-Bakul ini dirancang untuk memudahkan pengguna dalam hal produsen, reseller dan konsumen untuk melakukan transaksi secara online. Hingga menghasilkan output berupa bukti pesanan dan meminimalisir kesalahan perhitungan dalam proses transaksi juga memudahkan interasi produsen dengan reseller maupun konsumen.

3. Oleh Sari Anggarawati, Renaldi Prayoga, Dyah Budibruri Wibaningwati dan Anak Agung Eka Suwarnata, (2021). Agribisnis, Fakultas Pertanian, Universitas Nusa Bangsa. Pemasaran Produk Sayur Kelompok Wanita Tani Kecamatan Tanah Sareal, Kota Bogor di Era Pandemi Covid-19. Masalah pada penelitian ini adalah pem asaran produk sayur menjadi peluang batas holtikultura kelompok wanita tani kecamatan tanah sareal, kota Bogor di era pandemic covid-19. Metode yang digunakan dalam penelitian ini yaitu metode pendekatan deskritif kualitatif. Hasil penelitian dapat ditarik kesimpulan bahwa ketahanan pangan dan pertanian kota bogor mendorong petani beralih ke

konsep perkotaan. Konsep tersebut di nilai lebih efesien meningkatkan produksi pertanian, yaitu sistem hidroponik dan aquaponik.

1. Oleh Wahyuddin, Syahirun Alam dan Imran Rosadi Said, (2021). Informatika, Universitas Muhammadiyah Parepare, Indonesia. E-Commerce Bumbu Masakan KWT (Kelompok Wanita Tani) Setia Desa Pakkodi Kabupaten Enrekang. Masalah pada penelitian ini pemasaran produk. Sistem produksi yang dilakukan bersifat pesanan. Jika mereka mampu mencari pedagang pengumpul dan relasi, dapat memperoleh pesanan yang banyak dan kelemahan sistem pemesanan ini waktu peputaran uang yang cukup lama. Metode yang digunakan yaitu metode survei dengan cara mengumpulkan data atau informasi tentang populasi dengan menggunakan sample yang relative kecil, pengujian menggunakan black box dan white box. Hasil dari penelitian ini dapat di Tarik kesimpulan telah dihasilkan suatu sistem E-commerce Bumbu masakan Kelompok Wanita Tani Setia di Desa Pakkodi-Kabupaten Enrekang, dan sistem yang dibangun dapat menampilkan seluruh produk Kelompok Wanita Tani. Sehingga dapat lebih mudah untuk mempromosikan dan menjual produk Bumbu masakan Kelompok Wanita Tani.